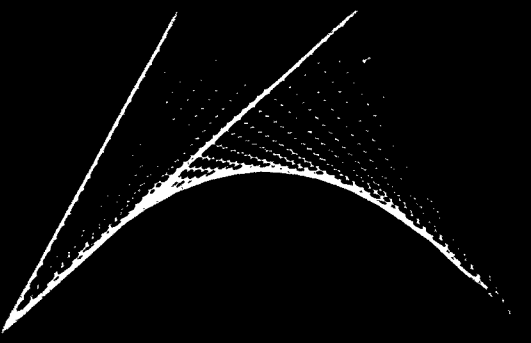

TM-WD-555/304/00

Defense Atomic Support Agency
Department of Defense Damage Assessment Center
Initial System

1604 JOVIAL Compiler
Vol. V: Description of System-Dependent Procedures

1 September 1962

TM-WD-555/304/00

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract DA-49-146-XZ-070
at the Washington Division / 5821 Columbia Pike / Falls Church, Virginia.

1604 JOVIAL Compiler

Description of System-Dependent Procedures

by

System Support Group

Washington Division

1 September 1962

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

(Page 2 Blank)

TABLE OF CONTENTS

(Page 4 Blank)

INTRODUCTION

The areas of the 1604 JOVIAL compiler described in this document are isolated because of their unique machine (configuration) and system dependence. These areas undergo the most drastic changes when the compiler is transferred from the DASA program production system to any other system.

The first chapter describes the routines that provide compool sensitivity. These routines expect the compool to be in a specific format. Furthermore, some of the rules that govern these routines (e.g., all compool tables are considered rigid length) are germane only to the production of programs for the DASA system.

The second chapter describes the routines that process direct code. These routines reflect the particular assembler (modified CODAP ϕ) and the preprocessor that are used to produce programs for the DASA system.

The third chapter describes the routines that process input and output statements. These routines provide input/output capability on the DASA machine configuration and are therefore machine configuration-dependent.

These descriptions are designed to aid the programmer who is responsible for maintaining the compiler. They should be used only in conjunction with a listing of the related JOVIAL statements and with the documents that describe the phase (translator or generator) of the compiler in which these routines are housed.

CHAPTER 1

PROCESSING OF COMPOOL IDENTIFIERS

1.0 INTRODUCTION

Compool sensitivity for the 1604 JOVIAL compiler is made possible by the inclusion of three procedures, RPOOL, POOL, and CONVB, within Pass 1 of the Generator. RPOOL deals with the compool in its entirety. POOL operates within individual sections of the compool. CONVB is used to convert characters from one code to another.

1.1 RPOOL

The function of RPOOL is to determine if the object program requires a compool; to read the required compool from tape into core; and to indicate the presence or absence of a compool. RPOOL has no input or output parameters or local variables.

Description. The procedure begins by checking item COMPOL to determine if a compool is requested by the control card of the object program. If RPOOL finds that a compool is not needed, RPOOL returns to the main program. If a compool is needed, it is read from tape into core. Because the sections of the compool are in one continuous tape record, and because the variable length tables (one per section) must be read into fixed, maximum-length, non-contiguous core locations, RPOOL is written in direct code.

To read and place compool tables in specified areas of core, RPOOL monitors the input for each desired compool section. Each section is identified by a table name and total number of words. If the input does not contain one of the sections listed within RPOOL, that entire compool section is read into one word of core. If the section is needed, its location and total number of words are placed in the I/O control word and the section is read into its proper core location. This process continues until the end-of-record is sensed. If a parity is sensed, the tape is backspaced and the entire process is repeated. When no parity is sensed, the item COMPOL is set to indicate that the compool is present.

1.2 POOL

This procedure searches a compool[1] section for a given identifier. If the identifier is found, POOL sets specific items in the Generator equal to the given compool information. If the identifier is not found, an exit from POOL

1 The compool used is described in SDC Field Note FN-WD-5545/513/00, 12 February 1962.

is made without any items being set. POOL is entered from IDEAL[2]. Only express items are used as its input and output parameters.

Local Variables.

RAY - a simple item indicating an array item.

XX - a simple item for processing one byte.

AA - a simple item for indexing within compool item CISYM.

BB - a simple item for indexing the compool status table CSTA.

NS - a simple item for indexing the number of states of a status item.

DD - a simple item for indexing the DICT table.

RR - a simple item for indexing the mixed part of the compool item table CITM.

SS - a simple item indexing the item table where the item's associated table entry is given.

POOLA - a simple item containing the identifier in BCD.

POOLAX - a nine-character simple item overlaid with POOLA.

BCDSTC - used to convert statuses from BCD to STC.

Description. POOL is called by IDEAL to search a compool section for an identifier when IDEAL has failed to locate it in the dictionary. POOL first checks NCHAR, and if NCHAR is found to be greater than five, POOL returns to IDEAL (because compool identifiers are limited to no more than five characters).

The requested identifier in IDNT is moved to POOLA and converted from STC code to BCD. POOLA is then used to determine the identifier's presence in or absence from the compool.

POOL considers two conditions when called by IDEAL. Item PARAM, which is set by IDEAL, indicates these conditions and controls what will be done. If PARAM is not equal to one or five, POOL returns to IDEAL. If PARAM equals one, indicating that a designation label has been encountered in the program, POOL searches the program section of the compool (CPRG) for the identifier. If it is found, the dictionary item DEFN is set to two, to indicate that the item is

2 SDC Technical Memorandum, TM-555/020/00, 1 July 1962

compool-defined. Also, CLAS is set to 12, which indicates a compool-defined program. If the identifier is not found in the CPRG section of the compool, POOL returns to IDEAL.

If PARAM equals five, which indicates that a variable is being processed, NCHAR is checked first. If NCHAR is not equal to four or five, POOL returns to IDEAL. If NCHAR equals four, it is necessary to search only the table section of the compool (CTBL), because table names in the compool have four characters. If the four-character identifier is not found, POOL exits to IDEAL.

If the identifier is found, the following dictionary items are set:

DEFN - set to two, indicating compool-defined.

CLAS - set to seven, indicating a table.

PACK - set to one, indicating the table is packed.

Other items are set according to the table definition information in compool item CTTYP. If CTTYP does not equal one, two, or three, Error Message 43 (identifier usage is not compatible with its class) is output and POOL returns to IDEAL.

If CTTYP equals one (parallel table):

TFRM is set to one, indicating a parallel table.

SIZE is set to the number of words per entry (CTMAX/CTWDS, which is the number of words in the table divided by the number of entries).

CHL1 is set to the number of entries (CTWDS).

POOL then returns to IDEAL.

If CTTYP equals two or three (serial or mixed table):

TFRM is set to two to indicate a serial table.

PDAT is set to CTVAR, which indicates rigid or variable table length (TRUE equals variable).

SIZE is set to the number of words per entry (CTWDS).

CHL1 is set to the number of entries (CTMAX/CTWDS).

POOL then returns to IDEAL.

If POOL finds NCHAR equal to five, it searches table CITM for the desired item name. It may find that the item name is a subordinate of a table, an array, or even another item. If the desired identifier is not encountered by the search of table CITM, POOL returns to IDEAL. Once the desired identifier is found, two referencing counters (RR and SS) are set. RR is set equal to the entry number in the second half of the CITM table, where the variable information describing the compool item is located. SS is set equal to RR. If the desired item is subordinate to another item, SS is modified to contain the entry number in the second half of the CITM table, where the cross-reference entry number to the CTBL table is located. If the item requested by IDEAL is subordinate to an array, an indicator is set. By use of reference counter RR, the following dictionary items are always set:

CLAS set to five (table item).

FPNM set to CIWRD (word in the entry where located).

DEFN set to two (compool-defined).

PACK set to one (indicates packed table).

FBIT set to CIBEG (beginning bit position of word).

If CICOD equals one (fixed point):

TYPV set to one (fixed point).

BRGT set to CIFRA (number of bits right or fractional bits).

SIZE set to CIBTS (number of bits).

PDAT set to CIGND (signed or unsigned).

If CICOD equals two (Boolean):

TYPV set to two.

SIZE set to one (number of bits).

If CICOD equals six (integer) or 15 (X-coded):

TYPV set to six.

SIZE set to CIBTS (number of bits).

If CICOD equals four (floating):

TYPV set to four.

PDAT set to SIGND (sign).

SIZE set to 48 (number of bits).

If CICOD equals five (Hollerith):

TYPV set to five.

SIZE set to CIBTS (number of bytes).

If CICOD equals nine (transmission):

TYPV set to nine.

SIZE set to CIBTS (number of bytes).

If CICOD equals seven (status):

TYPV set to seven.

SIZE set to CIBTS.

BRGT set to NSTAT (index to VALU).

Status item statuses must be moved from the compool to the compiler table VALU. This is done in the area labeled A1 to H1.

For the above items which will have a CLAS set to five, CHL2 is set to the dictionary channel where the table description containing this item can be found. CHL1 will contain the number of words per entry.

If the item belongs to a table, the table name is moved into IDNT, NCHAR set to four, and the table section of the compool is then searched. Thus, the table is entered in the dictionary if it has not been previously entered. If the item belongs to an array, the dimensions of the array are stored in the DISH table and POOL returns to IDEAL. It should be noted, however, that the use of arrays by POOL is not fully implemented.

1.3 CONVB

The function of this procedure is to locate an identifier for conversion from STC to BCD or BCD to STC. CONVB has no local variables. It is entered from POOL.

Description. The identifier to be converted must be located in the item IDNT, with TEXT set to indicate the first byte and NCHAR set to the number of bytes to be converted. CONVB sets KONVR8, KONVR9, and KNV10 for the CLOSE routine KONVRT to convert the bytes from the input type given in DUM3 to the output type requested in DUM4 in the item IDNT itself. CONVB then calls KONVRT to do the actual conversion. KONVRT is a small direct code routine which converts bytes of information from STC to BCD and BCD to STC.

CHAPTER 2

PROCESSING OF DIRECT CODE

2.0 INTRODUCTION

The processing of direct code for the 1604 is performed by the JOVIAL compiler using Procedure Direct Code (DC), which is part of Translator Pass 1 (T1). DC processes direct code cards after Generator Passes 1 and 2 have completed their operations so that information found in the direct code can be added to the table DICT and its associated tables before the translation begins. Procedure DC allows the object program to cross-reference direct code labels with the JOVIAL identifiers or compool identifiers. If the direct code is contained within a procedure, local cross-referencing is first attempted to establish the direct code with labels in the procedure. If this cannot be done, express cross-referencing is attempted to relate the direct code with labels in the main program. Cross-referencing from one procedure to another must never be done. If the direct code is not contained within a procedure, cross-referencing can be only express. All mnemonic or octal codes may be used, as may a set of pseudo-operation codes that include preprocessor codes.

2.1 GENERATOR PASS 1 (G1) PROCESSING

After encountering the DIRECT bracket, Generator Pass 1 (G1) scans each direct code card, searching for an ASSIGN operator or the terminating bracket, JOVIAL. All direct code cards are counted as encountered, one through n. To distinguish between the same local and express identifiers when direct code is encountered in a procedure, G1 collects pertinent information in the PDRT table which is used by the DC procedure in T1. The four items in PDRT which permit this are:

PDRC - the beginning dictionary channel of the procedure of which this direct code is a part.

PDRL - the terminating dictionary channel of the procedure.

PDRN - the number of direct code cards which belong to this procedure.

PDRF - the starting sequential card count when direct code in this procedure is encountered.

To permit referencing JOVIAL-defined identifiers, i.e., those defined outside the scope of direct code, the DISH table is passed on to T1 via tape. The DISH table represents all of the identifiers (the numeric HASHed values of the identifiers) and contains for each HASHed identifier an index to the dictionary. The dictionary items FCHB and NCHB are used to index the IDNS string to obtain the actual identifier. G1 encodes all encountered direct code cards in BCD and writes them onto the tape declared DIRCT.

2.2 DC PROCESSING

DC produces a set of variables in the table SKDUE for each direct code card and then writes SKDUE onto a tape, SKDRCT, for input to T2. DC is entered from the main program of T1. It has no input or output parameters.

Local Variables.

DD - DICT entry

YY - Byte of the card image

ZZ - Number of bytes

JOC - Converted decimal number

FLOAT - Floating point mterm

NNN - Converted octal number

SL - Byte of the starting letter in mterm or LOC field

NL - Number of letters

SN - Sign of the number in mterm

BN - Byte of the first digit in mterm

NON - Number of numbers in the mterm

EN - Type of character at the end of the number

FP - First byte of a floating point number

FN - Number of digits in the floating point number

SD - Sign of the digits after the number

FD - First byte of the digit in the power

ND - Number of digits in the power

NDA - Power of 10 or two

DO - Indicates an eight or nine in a set of numbers

SW2 - Zero = LOC field; one = mterm

XH - Transmission work area

XX - Byte work area

PRO - DICT entry when a procedure begins

SWN - Presence of a number in mterm

ENI - NOP

NCBN - First byte in the B power of a DEC card

NC - Number of bytes

FC - Reference to self, indicated by a (*)

HOPN - A subscripted item containing the CODAP symbolic codes

Pseudomnemonic Codes. The following list of codes are allowed by the direct code procedure:

POSA	SEN	REM	IDENT
POSQ	SEL	EQU	FINIS
RESA	ACT	ORG	END
RESQ	QRJ	DEC	BSS
LDQM	ARJ	OCT	BES
ENIL	SRZ	BCD	BTS
INIL	RTJ	FLX	LIB
MASK	NOP	TEL	

Description. Direct code cards are read and processed one at a time in four distinct segments: the location field (columns 1-8 of the card); the operation code field (columns 10-13); the index, or bterm, field (columns 17-18); and the mterm field (columns 20-39).

The main body of this procedure is oriented toward input, output, and CODAP-compatible legality checking. Calls are made to the following procedures and CLOSE routines which perform the actual formulation of the complete location and mterm fields:

CLOSE NUMBER is called to form the labels for routine RØ to process.

CLOSE OTOB is called to convert an octal STC number to binary.

CLOSE DTOB is called to convert a decimal STC number to binary.

Procedure TRANS[1] is called to convert the first 40 card columns from BCD to STC.

Procedure CONV is called to convert character coding from one code to another (see below).

The CODAP pseudocodes RST and WST are not implemented. CODAP pseudo-operation codes DEC, OCT, BCD, FLX and TEL are implemented but are restricted to one word of information per direct code card. When identifiers (labels, tags) are present either in the location or mterm fields, CLOSE RØ calls procedures HASH[2] and SRCH[2] to determine if the identifier had been previously used.

The area in CLOSE RØ from tag RØ1 to R5 insures proper compatible usage and referencing of identifiers within the scope of direct code. Further, this area provides that cross-referencing within a procedure will be first local and then express, but never from one procedure to another. Referencing of arrays, switches, files, procedures, closes, and strings in direct code is illegal. Compool-defined identifiers previously referenced in a JOVIAL statement or those which are unique (i.e., not defined by a label used elsewhere in direct code) will be permitted in this region. If an identifier has not been previously defined, or if it exists in a main program or in another procedure, etc., an entry is made in the DICT table for further referencing.

2.3 CONV

The function of procedure CONV is to convert a given number of bytes from any one to another of six types of character coding (BIN, BCD, STC, FLX, TEL, or SPFXL). CONV is entered from procedure DC.

Description. The information to be converted is located in the CARD image. The formal parameter DUM1 contains the first column, and DUM2 contains the number of columns to be converted. CONV converts the bytes from the input type given in DUM3 to the output type requested in DUM4 in the CARD image itself. A table look-up substitutes the requested type for the input type in the CARD image.

2.4 ERROR MESSAGES

Number	Explanation
201	Mterm contains more than five digits.

1 SDC Technical Memorandum, TM-555/302/00, 1 June 1962

2 SDC Technical Memorandum, TM-555/020/00, 10 July 1962

(Page 16 Blank)

Number	Explanation
203	More than eight characters between two minus zero symbols (ō) on a FLX or TEL card.
204	DEC is illegal: too many digits, non-numeric characters.
205	BSS is illegal: non-numeric characters in mterm.
206	OCT contains more than 16 digits and/or an eight or nine in mterm.
207	The LOCN or mterm has more than six characters (symbolic).
208	Duplicate tags.

CHAPTER 3

PROCESSING OF INPUT/OUTPUT STATEMENTS

3.0 INTRODUCTION

Several procedures and closed statements in T2 are employed to generate instructions which will accomplish input/output operations. The particular input and output operations are stated in the JOVIAL language. These statements are encoded by G1 and T1 and passed to T2 in the Intermediate Language table and the dictionary. The procedures and closed statements in T2 use this information to generate either the specific external function codes or a calling sequence to a library routine which will accomplish the input/output operations. The required library routines are incorporated with the compiled program during assembly.

3.1 ACCCK

This closed routine is used to check for a subscript or bead in the accumulator when input and output statements are processed. If ACCCK finds either in the accumulator, it sets up instructions to store the subscript or bead in temporary storage. The program then exits from ACCCK.

The temporary storage assignments for a subscript or bead are:

If the operand has only one subscript, the subscript is stored in TEMPS+1.

If the first of the operand's two subscripts is found in the accumulator, it is stored in TEMPS+2.

If either the second subscript or the first or second bead is found in the accumulator, it is stored in TEMPS+1.

3.2 CENTSZ

This closed routine is used in processing input and output statements whose dataname is a table. CENTSZ checks for a parallel table with more than one word per entry. If one is found, an error message is written on the debug tape.

3.3 CALL

This procedure is used to prepare the first word of the calling sequence for output library routines.

Description. If the output hardware is one of those listed below, CALL sets item CELL1 equal to zero as an error flag. If the output hardware is not one of those listed below, CELL1 is set equal to one and an error message is printed on the debug tape.

CALL prepares instructions to call the appropriate library routine, using BLIB[1], which actually generates the RTJ. CALL also sets an indicator for the next instruction to be forced into an upper location.

1. Magnetic Tape. If a binary file is to be written on magnetic tape, a jump to library routine CRITB' is generated. If the file is decimal, CRITD' is called. CALL prepares the next instruction, which indicates the channel and cabinet number in the operation field, and the tape number in the bterm field.

2. Typewriter. A jump to CWTYP' is generated. No lower instruction is prepared.

3. Paper Tape. A jump to CPTPN' is generated. If the file is binary, the mterm of the next instruction is set equal to zero. If it is decimal, the mterm is set equal to one.

4. Card Punch. A jump to CPNCH' is generated. The mterm of the next instruction is set to zero if the file is decimal, or to one if the file is binary.

5. Printer. A jump to CPRNT' is generated. No lower instruction is prepared.

3.4 CBRGT

Procedure CBRGT is used in processing input statements to set three items for magnetic tape assignments:

COPN is set to an octal value equal to the channel number times eight plus the 1607 unit number.

CBTERM is set to the tape unit number.

CNTRL is set to a binary value equal to the decimal value of the hardware identifier ("T" number).

1 SDC Technical Memorandum, TM-WD-555/303/00 (to be published October 1962)

The item settings for each possible magnetic tape are as follows:

Hardware Identifier	BRGT (Octal)	CBRGT Setting		
		COPN (Octal)	CBTERM	CNTRL (Decimal)
T01	11	12	1	1
T02	12	12	2	2
T03	13	12	3	3
T04	14	12	4	4
T05	31	32	1	5
T06	32	32	2	6
T07	33	32	3	7
T08	34	32	4	8
T09	51	52	1	9
T10	52	52	2	10
T11	53	52	3	11
T12	54	52	4	12

Description. CBRGT checks the value of BRGT subscripted by CSUB to determine the setting of items COPN, CBTERM, and CNTRL. When they are set, the program exits from CBRGT.

An express variable, CSUB, is used by CBRGT as an input parameter. CSUB is set to PILF or DILF (1), depending on whether the source of the required information is an IL operator or operand.

3.5 CINPT

This procedure is used in processing all input statements. The calling sequences for the appropriate library routine are generated, as well as any instructions required for manipulation of the input data.

Description. Before processing any input statements, CINPT performs the following four steps:

1. Unpacks the IL table into the DIL, SBIL, BIL1, and BIL2 tables;
2. Unpacks the dictionary items for the first operand;
3. Checks the contents of the accumulator for a subscript or bead, and
4. Sets an index register equal to DILF(1) to save time and space.

Calling sequences vary as follows with the input hardware to be used:

1. Magnetic Tape. Initial parameters are established and instructions are generated to set a flag to zero to indicate that this file is to be used for input.

   a. If the dataname is a table modified by NENT, instructions are generated for a jump to library routine CREDB' and to set an indicator for the next instruction to be forced into an upper location.

   b. If the dataname is a table not modified by NENT, instructions are generated to store the proper starting and terminating (terminal address plus one) addresses of the dataname in the calling sequence.

      1) If there was an error in procedure CIOTBL, the error indicator item, ERR, is set equal to zero and an exit is made from the routine.

      2) If there was no error in CIOTBL, a jump to library routine CREDB' is generated and an indicator is set for the next instruction to be forced into an upper location.

         a) If the file is binary, a jump to library routine CREDB' is generated.

         b) If the file is decimal, a jump to library routine CREDD' is generated.

   c. If the dataname is subscripted by a bead, an error message is written on the debug tape and an exit made from the routine.

   d. If the dataname is a table item with a single subscript, CINPT uses procedure CTBLI to prepare instructions for storing the proper starting and terminal-plus-one addresses in the calling sequence.

   If the input hardware is not magnetic tape, item BBUN is set to hardware type and then used by switch IOUNIT to determine the calling sequence for the proper library routine.

2. Clock. The contents of the clock are placed in the accumulator.

   a. If the dataname is a simple item, the accumulator is stored in the proper item and an exit is made from the routine.

b. If the dataname is a subscripted item, CINPT uses procedure DEPOS[1] to deposit the accumulator in the proper location, and an exit is made from the routine.

c. If the dataname is neither a simple nor subscripted item, an error message is written on the debug tape and an exit is made from the routine.

3. Typewriter. A jump to library routine CRTYP' is generated and an indicator is set for the next instruction to be forced into an upper location.

4. Paper Tape. A jump to library routine CPTRD' is generated and an indicator is set for the next instruction to be forced into an upper location. The mterm of the next instruction is set to one or zero, depending on whether the file is Hollerith or binary.

5. Card Reader. A jump to library routine CREAD' is called and an indicator is set for the next instruction to be forced into an upper location. The mterm of the next instruction is set to one or zero, depending on whether the file is Hollerith or binary.

a. If the input hardware is the secondary read station, the mterm of the next instruction is increased by two.

b. If the input hardware is not the secondary read station, the mterm of the next instruction is increased by one.

If the input hardware is other than magnetic tape, clock, typewriter, paper tape or card reader, an error message is written on the debug tape and an exit made from the routine.

Up to this point, CINPT has generated the first two half-words of the calling sequence and also instructions that will modify the calling sequence during execute time. The following description relates to that part of CINPT which generates the last two half-words of the calling sequence:

1. Table.

a. If the table is modified by NENT, the address of tablename minus one is set up in the mterm of the next instruction as the starting address of the input data. The address of the tablename is set up in the lower mterm as the terminating address.

1 SDC Technical Memorandum, TM-WD-555/303/00 (to be published October 1962)

b. If the table is not modified by NENT, two instructions of zeros are prepared so that the starting address may be stored in the upper mterm of this word and the terminating address may be stored in the lower mterm.

An exit is made from CINPT.

2. Beaded Item. If the item is modified by BIT or BYTE, an error message is written on the debug tape, and an exit is made from CINPT.

3. Simple Item. If the dataname is not subscripted, it is a simple item. Procedure CNOSU is used to set the starting and terminating addresses in the next word. An exit is made from CINPT.

4. Subscripted Item. Procedure CTBLI is called to prepare instructions to generate and store the starting and terminating addresses of the input data in the calling sequence.

   a. If item OPS is equal to five, two instructions of zeros are prepared for the calling sequence to contain the starting and terminating addresses of the data transfer.

   b. If item OPS is zero, a call to library routine CBUFR' (working storage space) is generated.

   c. If the item is not Hollerith or STC, or is less than or equal to eight literal characters, the mterm of the next two instructions is set to CBUFR' and CBUFR'+1 as starting and terminating addresses. Procedure BXODS[1] is used to set up the DIL and DICT tables so that the second operand will be the accumulator in an ASSIGN statement. The contents of CBUFR' are placed in the accumulator, and procedure DEPOS is used to generate instructions to properly position and store the input data.

5. Subscripted Literal Item Greater Than Eight Characters. To determine the starting address of the input data, the number of characters is divided by eight. If it is equally divisible, the number is reduced by one. The starting address for input data is set at CBUFR' minus the number calculated, and the terminating address is set equal to CBUFR'+1. Item HILL is set equal to two, indicating to procedure BORB[1] that the information to be stored

1 SDC Technical Memorandum, TM-WD-555/303/00 (to be published October 1962)

in the dataname is in the CBUFR' table. BORB is used to deposit the input data in the location specified by the dataname. An exit is then made from the routine.

3.6 COUTT

Procedure COUTT is primarily concerned with the second word of the calling sequence; that is, with the starting and terminating addresses of the data to be output. COUTT uses procedure CALL to generate the call for the appropriate library routine.

Description. Before processing any OUTPUT statements, COUTT performs the following four steps:

1. Unpacks the IL table into the DIL, SBIL, BIL1, and BIL2 tables;
2. Unpacks the dictionary items for the first operand;
3. Checks the contents of the accumulator for a subscript or bead, and
4. Sets an index register equal to DILF(1) to save time and space.

If the output hardware is:

1. Magnetic Tape. Initial parameters are established and instructions are generated to set a flag to one to indicate that this file is to be used for output.
2. Clock (CLOCKO). If the dataname is a simple item or table item, procedure ELDOD[1] is called to generate instructions to load and right-justify the item. Instructions are generated to store the accumulator in the clock and an exit is made from COUTT. If the dataname is not a simple item or a table item, an exit is made from COUTT.

Dataname instructions are prepared as follows:

1. Table.
   a. If the table is modified by NENT, procedure CALL is used to generate the call to the appropriate output library routine. If the error indicator CELL1 is set when returning from CALL, an exit is made from COUTT. If it is not set, the starting

1 SDC Technical Memorandum, TM-WD-555/303/00 (to be published October 1962)

address is set equal to the tablename minus one, the terminating address is set equal to the tablename, and both are placed in the calling sequence.

b. If the table is not modified by NENT, procedure CIOTBL is used to generate instructions to store the starting and terminating addresses in the calling sequence. If the error indicator ERR is set when returning from CIOTBL, ERR is set equal to zero and an exit is made from COUTT. If ERR is not set, procedure CALL is used to generate the call to the appropriate output library routine. If the error indicator CELL1 is set when returning from CALL, an exit is made from COUTT. If CELL1 is not set, one word of zeros is generated to receive the starting and terminating addresses of the table.

An exit is made from COUTT.

2. Beaded Item. If the item is modified by a BIT or BYTE, an error message is written on the debug tape, and an exit is made from COUTT.

3. Simple Item. If the dataname is not subscripted, it is a simple item. Procedure CALL is used to generate the call to the appropriate library routine. If CELL1 is set when returning from CALL, an exit is made from COUTT. If CELL1 is not set, procedure CNOSU is used to set the starting and terminating addresses in the next word. An exit is then made from COUTT.

4. Subscripted Item. CTBLI is called to prepare instructions to generate and store the starting and terminating addresses of the output data in the calling sequence.

   a. If item OPS is equal to five when returning from CTBLI, which indicates that the instructions are generated and stored, CALL is used to generate the call to the appropriate output library routine. If the error indicator CELL1 is set when returning from CALL, an exit is made from COUTT. If CELL1 is not set, the next word of the calling sequence is set equal to zero. The upper mterm of the next word will receive the starting address and the lower mterm the terminating of the output data. An exit is made from COUTT.

   b. If item OPS is not set equal to five when a return is made from CTBLI, a call to library routine CBUFR' (working storage) is generated. If the output data is not a Hollerith or STC item, OP is set equal to one (first operand) and procedure ELDOD is used to set up instructions to load the subscripted

item into the accumulator. The procedure goes to area COUT5A to set up instructions to store the accumulator contents in CBUFR'.

c. If the output data is less than or equal to eight bytes, item OP is set equal to one (first operand) and item ODPOS is set equal to zero. Procedure LITHP[1] is used to prepare instructions to load and right-justify the literal item in the accumulator. The procedure goes to area COUT5A to set up instructions to store the accumulator contents in CBUFR'.

5. Subscripted Literal Item Greater Than Eight Characters. To determine the starting address, the number of characters is divided by eight and the quotient is stored in CSIZE. If it is equally divisible, the CSIZE is reduced by one. The indicator HILL is set equal to one as a flag to BORB that an output statement is being processed. BORB is used to prepare instructions to remove output data from the dataname to CBUFR'. CALL is used to prepare the first word of the calling sequence. If item CELL1 is set when returning from CALL, an exit is made from COUTT. If CELL1 is not set, the upper mterm of the next instruction (starting address) is set equal to CBUFR' minus CSIZE, and the lower mterm is set equal to CBUFR'+1 (terminating address). An exit is made from COUTT.

6. Error. If the dataname is illegal, an error message is written onto the debug tape and an exit is made from COUTT.

3.7 CIMSUB

This procedure is used by procedures CIOTBL and CTBLI to prepare instructions for loading the accumulator with the value of a specified subscript (variable plus or minus constant).

Description.

1. Item OPS is equal to one or three, depending on which subscript (the first or second) is to be loaded into the accumulator.

2. Single Letter Subscript. Procedure IHNDL[1] is used to prepare loading instructions to put the present value of the single letter subscript into the accumulator.

1 SDC Technical Memorandum TM-WD-555/303/00 (to be published October 1962)

3. NENT of a Table. Instructions are generated to load the accumulator with the contents of the table name minus one.

4. Neither Single Letter nor NENT of a Table. Instructions are generated to load the accumulator with the temporary register or the variable. If there is a constant associated with this variable, it is added to or subtracted from the contents of the accumulator. An exit is made from CINSUB.

3.8 CIOTBL

Procedures CINPT and COUTT use CIOTBL when the dataname portion of the input or output statement is a table. CIOTBL is primarily concerned with generating instructions to store the starting and terminating address of the transfer in the calling sequence.

Description. CIOTBL sets an index equal to DILF(1).

.1. Double-Subscripted Table. The closed routine CENTSZ is used to check for a parallel table with more than one word per entry. If the table is parallel and has more than one word per entry, an exit is made from CIOTBL.

Item OPS is set equal to three and procedure CINSUB is used to generate instructions to load the value of the second subscript into the accumulator. Instructions are generated to increase the second subscript by one and store it in TEMPS+1.

Item OPS is set equal to one and procedure CINSUB generates instructions to load the accumulator with the value of the first subscript. An instruction is prepared to store the first subscript into TEMPS+2.

If the table has only one word per entry, instructions are generated to load the accumulator with the value of the first subscript (TEMPS+2). If the table (serial) has more than one word per entry, instructions are generated to multiply the number of words per entry by the value of the first subscript. Instructions are generated to increase the contents of the accumulator (words per entry times the number of entries) by the starting address of the table and to store the result in the starting address portion of the appropriate input or output calling sequence.

If the table has only one word per entry, instructions are generated to load the accumulator with the value of the second subscript plus one (TEMPS+1). If the table (serial) has more than one word per entry, instructions are generated to multiply the

number of words per entry by the value of the second subscript plus one (TEMPS+1). Instructions are generated to increase the contents of the accumulator by the starting address of the table and to store the result in the terminating address portion of the appropriate input or output calling sequence

An exit is made from CIOTBL.

2. Single-Subscripted Table. CENTSZ is used to check for a parallel table with more than one word per entry. If the table is parallel with more than one word per entry, an exit is made from CIOTBL.

Item OPS is set equal to one and procedure CINSUB generates instructions to load the accumulator with the value of the subscript. Instructions are generated to multiply the subscript value by the number of words per entry, to increase the result by the starting address of the table, and then to store the final result in the starting address portion of the appropriate input or output calling sequence.

Instructions are generated to increase the starting address (now in the accumulator) by the number of words to be transferred and to store this value in the terminating address portion of the appropriate input or output calling sequence.

An exit is made from CIOTBL.

3. Table Without Subscript. Instructions are generated to load the starting address into the accumulator and then to store this value into starting address portion of the appropriate calling sequence.

If this table is a variable length serial or parallel (one word per entry) table, instructions are generated to multiply the number of words per entry by tablename minus one (NENT), to increase this result by the starting address, and to store the final result in the terminating address portion of the appropriate calling sequence.

If this table is a rigid length table or a variable length parallel table with more than one word per entry, the total number of words to be transferred (number of entries multiplied by the number of words per entry) are calculated. Instructions are generated to add this total to the accumulator and to store the result in the terminating portion of the appropriate library routine.

An exit is made from CIOTBL.

4. Error. If the tablename is not subscripted by zero, one, or two subscripts, an error message is written onto the debug tape and error indicator item ERR is set. An exit is made from CIOTBL.

3.9 CTBLI

This procedure is used by CINPT and COUTT to prepare instructions to generate the starting and terminating addresses of Hollerith or STC table items and to store them in the correct input or output calling sequence.

Description. CTBLI sets an index to value of DILF(1) to save space and time. If the item is not Hollerith or STC, or if it is Hollerith or STC but does not begin in byte zero and occupy an integral number of words, OPS is set to zero and an exit is made from CTBLI.

CINSUB is called to generate instructions to load the value of the subscript into the accumulator. If the item is located in a serial table, instructions are generated to multiply the accumulator by the number of words per entry. Instructions are generated to increase the accumulator by the starting address of the item's table and to store the result in the starting address portion of the calling sequence. Then the instructions are generated to increase the starting address (accumulator) by the number of words the item occupies and to store this result in the terminating address portion of the calling sequence. OPS is set to five and an exit is made from CTBLI.

3.10 CNOSU

This procedure prepares the starting and terminating addresses for the appropriate input or output library routine calling sequence when the dataname is a simple item or a constant.

Description. If the dataname is not a constant or a simple item, an error is written on the debug tape and an exit is made from CNOSU. The starting address is set with the address of the dataname. The terminating address is set to the address of the dataname plus one, except for literals. The terminating address for literals is set to the starting address plus the number of words which the literal occupies.

3.11 CRTJ

This procedure prepares a return jump to a library routine. OPN is set to RTJ, MTC is set to nine, and SKD00[1] is called. All other parameters for SKD00 are set before CRTJ is called.

---

1 SDC Technical Memorandum, TM-WD-555/302/00, 1 June 1962

3.12 CSET

This procedure prepares the second half of the first word of many of the calling sequences for the magnetic tape library routine. Only input channels are considered by this procedure. The operation code is set to the channel number times eight plus the 1607 unit number. The bterm is set to the tape unit number.

3.13 CSTAT

This procedure is used in conjunction with CSTUS to prepare instructions for library routine calling sequences (for library routines that check the status of magnetic tapes). CSTAT prepares instructions for placing tape identification information in the accumulator. This information is considered to be input data. If the file that is interrogated is an output file, the tape identification information is changed accordingly. This information is then shifted to the proper portion of the word to await placement in the calling sequence.

3.14 CSTUS

This procedure is used in conjunction with CSTAT to set up the instructions for calling sequences to library routines that check the status of magnetic tapes. CSTUS sets up an instruction to store the accumulator (which now contains the first word of the calling sequence) into the next location. CSTUS also reserves the word that is to receive the first word of the calling sequence.

3.15 I'O

This procedure handles the POS modifier, OPEN INPUT, OPEN OUTPUT, SHUT INPUT, SHUT OUTPUT and all status checks that involve peripheral equipment.

Description. The dictionary channel of the first operand is loaded into a subscript to increase the efficiency of the compiler.

1. POS Modifier. If the second operand is zero, CBRGT is called and then a jump to CRWND' is generated. CSET, which will set the lower half of the first word of the calling sequence, is called. If the file required is not the first file on the tape, the calling sequence for CSKPF' is generated.

   If the second operand is not zero, the calling sequence is generated for either CBACK' or CSKIP', depending on whether I'O was entered from JSUB (subtract operator) or from JADD (add operator).

   An exit is made from I'O.

2. OPEN INPUT.

   a. Tape. CBRGT is called and then instructions are generated to set hardware identifier in an indicator. The calling sequence for CRWND' is generated. If this is not the first file on the tape, the appropriate calling sequence for CSKPF' is generated. If a dataname is included in the statement, CINPT is called.

      An exit is made from I'O.

   b. Clock. Instructions are generated to clear and start the clock. If a dataname is included in the statement, CINPT is called.

      An exit is made from I'O.

3. OPEN OUTPUT.

   a. Tape. If this is the first time this file has been "opened" or if it is the first file on the tape, instructions are generated to set the hardware identifier in the proper indicator. The calling sequence for CRWND' is generated. If a dataname is included in the statement, COUTT is called.

      An exit is made from I'O.

   b. Printer. Instructions are generated for page ejection. If dataname is included in the statement, COUTT is called.

      An exit is made from I'O.

   c. Clock. If a dataname is not included in the statement, instructions are generated to clear and activate the clock. If dataname is included, COUTT is called and, upon return from COUTT, an instruction is generated to activate the clock.

      An exit is made from I'O.

4. SHUT INPUT. If a dataname is included in the statement, CINPT is called. If the hardware device is the clock, an instruction to stop the clock is generated.

   An exit is made from I'O.

5. SHUT OUTPUT. If a dataname is included in the statement, COUTT is called.

   a. Tape. The calling sequence for CWEOF' is generated and then an exit is made from I'O.

   b. Clock. An instruction is generated to stop the clock and then an exit is made from I'O.

6. Status Checks.

   a. Tape. When the check is for ready, parity or buffer length error, instructions are generated and CSTAT and CSTUS are called to obtain the calling sequences for CREDY', CPRTY', CWERR' and CBLTH'. The instructions that are generated compute the calling sequence to the appropriate library routine. When the check is for end-of-file or end-of-tape, no computation is required. CRTJ and CSET are used to generate the call for CENDF' and CENDT', respectively.

      Depending on whether EQ or NQ was specified in the statement, the logic of the next two words of the calling sequence will be reversed.

      An exit is made from I'O.

   b. Card Reader. Instructions are generated to exit "on ready" or "not ready," depending on whether "ready" or "end-of-cards" is specified in the statement. The true transfer is generated after considering the use of EQ or NQ in the statement.

      An exit is made from I'O.

   c. Printer and Punch. Instructions are generated to exit "on ready." Also, the true transfer is generated after considering the use of EQ or NQ in the statement.

      An exit is made from I'O.

   d. Paper Tape Reader and Punch. If NQ is specified in the statement, an exit is made from I'O. The true transfer is generated.

      An exit is made from I'O.

e. Typewriter. The appropriate instructions are generated to check for carriage return, lower case, or ready, depending on the statement. NQ is ignored when specified with ready, but for all other cases, the true transfer is generated.

An exit is made from I'O.

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
DEFENSE ATOMIC SUPPORT AGENCY -
DEPARTMENT OF DEFENSE - DAMAGE
ASSESSMENT CENTER - INITIAL SYSTEM -
1604 JOVIAL COMPILER - VOL. 5:
DESCRIPTION OF SYSTEM-DEPENDENT PROCEDURES.
Scientific rept., TM-WD-555/304/00, by
System Support Group, Washington Division.
1 September 1962, 32p.
(Contract DA-49-146-XZ-070)

DESCRIPTORS: Digital Computers.
Machine Translation.

Identifiers: JOVIAL. UNCLASSIFIED

UNCLASSIFIED

Describes routines that provide compool sensitivity, routines that process direct code, and routines that process input and output statements of the CDC-1604 computer JOVIAL compiler.

UNCLASSIFIED